

## REMARKS

Applicants respectfully request that the above-identified application be reexamined.

The Office Action mailed on May 5, 2004 ("Office Action"), rejected all the claims in the application. Claims 1, 3, 4, 7, 9, 13, 15, and 16 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Straub, U.S. Patent No. 5,430,878 (hereinafter "Straub") in view of Nowlin, Jr. et al., U.S. Patent No. 6,484,309 (hereinafter "Nowlin"). Claims 2, 5, 6, 8, 10, 11, 12, 14, 17, and 18 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Straub in view of Nowlin, Jr. et al., as applied to Claims 1, 7, 13, and 15, and in view of Preisler et al., U.S. Patent No. 5,675,803 (hereinafter "Preisler").

This amendment amends the claims to clarify the meaning of certain claim terms, which applicants believe were implicit in the claim language. For the reasons hereinafter set forth, applicants respectfully submit that the rejection of Claims 1-18, as amended, in view of the teachings of the cited references, should be withdrawn and this application be allowed.

Prior to discussing the reasons why applicants believe that the claims of this application are clearly allowable, a brief discussion of the present invention, followed by a brief discussion of the cited and applied references, are presented. The following discussions of applicants' invention and the cited and applied references are not provided to define the scope or interpretation of any of the claims of this application. Instead, these discussions are provided to help the United States Patent and Trademark Office better appreciate important claim distinctions discussed thereafter.

### Summary of the Invention

The present invention is directed to methods, computer-readable medium having computer-executable instructions, and systems for patching computer application programs that are not compatible with the computer operating system executing the computer application program. More specifically, the methods, the computer-readable medium having computer-executable instructions, and the systems determine whether or not the computer application program is compatible with the computer operating system executing the computer application program. If the computer application program is determined to be incompatible with the computer operating system, a debugger is started to run and patch the computer application program.

Application compatibility with the operating system in one embodiment of the invention is determined by checking application identity information against a database containing a list of

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

applications that are currently known to be incompatible with the operating system. If the computer application program is found in the database, a set of attributes is checked to determine if the particular version of the application is incompatible. If the checked attributes match the ones in the database, the application is determined to be incompatible with the operating system. If the application is not found in the database, the application is determined to be compatible with the operating system. Based on the determination, the application is either run with or without a debugger. Applications that work correctly under the operating system are not required to go through an additional level of execution created by the debugger application. Thus, such applications execute faster.

In one embodiment of the invention, if an application is found to be incompatible with the operating system, the operating system starts a debugger application that, in turn, runs the incompatible application. Before loading the incompatible application, the debugger loads a dynamic link library (DLL) that contains patches for the incompatible functions of the application. Specifically, the DLL contains a list of breakpoints specifying the location where the application needs to be patched, together with the appropriate code required to patch the application.

In this exemplary embodiment of the invention, when the debugger loads the DLL, a function is called that sets the breakpoints within the incompatible application. A breakpoint may be specified by (1) the module name and function name; (2) the module name and a memory offset; or (3) an address. The debugger also implements a handler that is capable of modifying the application code that is incompatible. In one embodiment of the invention, the debugger handler is capable of modifying the application code that is incompatible. In one embodiment of the invention, the debugger handler is capable of reading and writing memory within the incompatible application. Therefore, the handler is capable of modifying the incompatible code or inserting code into the application to fix the problem code contained within the application. For example, when the debugger reaches a breakpoint within the application, the debugger may be called to merely skip a portion of the incompatible code, or the handler may rewrite a portion of the code to contain a certain value in order to make the application compatible with the operating system.

One of the benefits of the use of a debugger application to patch incompatible applications is that this arrangement is very robust. The debugger is capable of monitoring every step an application takes while it is executing. Because the amount of code required to patch an application is generally very small, such as 200 bytes, patches are readily accessible to a user either through a Web site or FTP site.

## Summary of the Cited and Applied References

### Summary of Straub

Straub describes a method of examining and revising an image of a computer program so that the program can operate properly with the configuration of a computer. The method reads a computer application from a fixed storage medium. It then examines the program image to find whether the computer program recognizes the configuration of the computer it will run on. If it does not, the program image is revised so that it will recognize the configuration of the computer (first revision). If the computer program does recognize the configuration of the computer, then the method determines whether the program image can be operated from the lower memory area of the computer. If the program image cannot be so operated, the program image is revised so it can be run substantially anywhere in the computer's memory (second revision). If the program image can be run from the lower memory area of the computer, the computer operating system will execute the program.

The first revision is done by the computer checking a table maintained by the computer system. The table contains compatible computer program names and version numbers of operating systems that can run on the computer system. The computer references the table to obtain the operating system version number associated with the name of the computer program and stores this version number in the program image of the computer program.

The second revision is done by the computer replacing a predetermined set of instructions and data in the program image with another alternate set of instructions and data. This replacement enables the program image to be run from almost anywhere in the computer memory.

In summary, Straub merely provides a method for revising a computer program image in order for the computer program to be run on a particular computer configuration and anywhere in the computer memory. Straub teaches only two particular revisions—(1) storing operating system version number; and (2) replacing a predetermined set of instructions and data in the program image. Straub also teaches maintaining a table of compatible program names in the computer system. Straub does not teach or even remotely suggest using a debugger to execute a computer program (application), much less a debugger that sets breakpoints, runs an application, and patches the application at the breakpoints.

### Summary of Nowlin

Nowlin purportedly teaches a method that converts a computer program designed for one operating system to run on a second operating system without a recompilation. This invention is

specifically geared toward converting a non-Windows® CE application program into a Windows® CE-compatible computer program.

The method allegedly taught by Nowlin includes (a) loading a computer program on a computer system having a non-Windows® CE operating system; (b) translating the computer program to run on another computer system having a Windows® CE operating system; and (c) after the translation is completed, transferring the converted computer program to the second computer system running a Windows® CE operating system.

The translation is accomplished by creating a translation layer on the first computer system. The translation layer communicates with a non-Windows® CE system using a non-Windows® CE system calling convention, and with a Windows® CE system using a Windows® CE system calling convention. The translation layer converts calls for a non-Windows® CE operating system to calls that are compatible with a Windows® CE operating system. The translation includes parsing executable and dynamic link libraries and using separate filters to translate each of these different file types. The translation includes modifying the header of a file so that a Windows® CE operating system recognizes a file as a valid Windows® CE file.

In summary, Nowlin merely provides a mechanism of parsing a non-Windows® CE-compatible computer application and translating the necessary program code on the non-Windows® CE computer system so that the application can be run on a Windows® CE computer system without recompilation. Nowlin does not teach or even remotely suggest using a debugger to execute a computer program (application), much less a debugger that sets breakpoints, runs an application, and patches the application at the breakpoints.

#### Summary of Preisler

Preisler describes a runtime debug operation designated "Fix and Continue" operation. Preisler permits a user to begin a debugging session if an error is encountered in executable code. The user edits the corresponding source code to correct the error and then executes a "Fix and Continue" command without leaving the debugging session. The "Fix and Continue" command calls the compiler to recompile a source code file with the newly edited text, receives the resulting recompiled object code file from the compiler, uses a dynamic linker to link the recompiled object code into the target application program process, patches the previous versions of the same object code file to refer to the newly compiled code, resets any required variables and registers, and resets the program counter to the line of code being executed to where the error was discovered. The debugger then continues the debug session, thereby saving the time it

would ordinarily take to quit the debug session, relink, and reload the target program, and start the debug session again.

Preisler's "Fix and Continue" command merely provides for pausing the debugging of a program to allow a programmer to add source code without leaving the debugger. Nowhere does Preisler teach or even remotely suggest a functionality of determining whether a computer application is compatible with the operating system executing the computer application. Nor does Preisler teach or even remotely suggest a debugger that sets breakpoints, runs an application and patches the application at the breakpoints.

### The Claims Distinguished

The Office Action has failed to show, and the applicants are unable to find, where any of the cited and applied references, alone or in combination, disclose, teach or suggest the subject matter of the claimed invention. As the above summaries show, among other differences, none of the cited and applied references teaches, discloses, or suggests a debugger that runs an application that is incompatible with the operating system the application is on. Nor does any of the cited and applied references teach, disclose, or suggest a debugger that sets breakpoints, runs an application and patches the application at the breakpoints.

Prior to discussing in detail the reasons why applicants believe that the claims of this application, particularly as amended, are clearly allowable, attention is again directed to the law relating to combinations of references.

The Office Action recognizes that Straub, Nowlin, and Preisler alone fail to teach all of the elements of Claims 1-18. The Office Action makes various assertions (discussed in detail below) that it would be obvious to one of ordinary skill in the art to combine the teachings of Straub, Nowlin, and Preisler. Without addressing the accuracy of individual assertions here, applicants submit that the **assertions** in the Office Action **regarding the obviousness** of combining the teachings of the cited references **are based on impermissible hindsight construction of the claimed invention**. Straub is directed to revising a computer program image by storing an operation system version number and replacing a predetermined set of instruction and data in the computer program image. Nowlin is directed to converting a non-Windows® CE-compatible computer application and translating the necessary program code so the application can run on a Windows® CE computer system without recompilation. Preisler, on the other hand, is directed to pausing a debugging session to allow a programmer to correct the error without leaving the debugging session. Applicants submit that one of ordinary skill in the art would not be motivated to combine the Preisler system with Straub and Nowlin as both

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

Straub and Nowlin apply to modifying a computer application program without the use of a debugger and there is no suggestion in either Straub or Nowlin of the need to use a debugger. Accordingly, applicants submit that there is no motivation for one of ordinary skill in the art to combine the teaching of these references.

Applicants further submit that the rejections of Claims 1-18 are predicated on combining prior art references that contain no teaching or suggestion of how the teachings of the cited references could be combined in any manner, much less the manner recited in the rejected claims. The claimed subject matter is taught only by the present application, not by the references. The Office Action fails to point out any teaching or suggestion in the references related to the desirability of combining their individual teachings. The rejections use hindsight reasoning based on the present disclosure to "produce" the claimed invention. Nor is there any obvious or apparent reason for combining the teachings of such disparate references. The references do not teach or suggest how they could be combined in any manner, much less the manner recited in the rejected independent claims (1, 7, 13). In this regard, attention is directed to the following Federal Circuit and C.C.P.A. decisions:

The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438, (Fed. Cir. 1991).

\*\*\*\*\*

It is wrong to use the patent in suit as a guide through the maze of prior art references, combining the right references in the right way so as to achieve the result of the claims in suit. Monday morning quarterbacking is quite improper when resolving the question of nonobviousness in a court of law. *Orthopedic Equipment, Inc. v. United States*, 217 U.S.P.Q. 193, 199 (Fed. Cir. 1983).

\*\*\*\*\*

Obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under Section 103, teachings of references can be combined **only** if there is some suggestion or incentive to do so.

*ACS Hospital Systems, Inc. v. Montefiore Hospital*, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984). (Emphasis added.)

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

The *ACS Hospital Systems, Inc. v. Montefiore Hospital* decision has been cited with approval by the Federal Circuit. See *In re Geiger*, 2 U.S.P.Q. 2d 1276, 1278 (Fed. Cir. 1987). Similar statements have been made in many decisions of the Board of Appeals.

Nor do we see any suggestion in either of the references which would lead anyone having ordinary skill in the art to combine the structure taught by either reference with that taught by the other.

In order to justify a combination of references such as is here suggested it is necessary not only that it be physically possible to combine them, but the art should contain something to suggest the desirability of doing so. Since the art does not suggest the use of either of the patented devices for . . . there is nothing to indicate that one should be modified in view of the other for that purpose.

*Ex parte Walker*, 135 U.S.P.Q. 195, 196 (Bd. App. 1962).

We have studied the references and the manner in which the examiner proposes to combine their teachings but we are unable to find in these references any suggestion that they should or could be combined, absent appellant's disclosure in the present application.

*Ex parte Lennox*, 144 U.S.P.Q. 224, 225 (Bd. App. 1964).

While as an abstract proposition it might be possible to select features from the secondary references, as the examiner has done, and mechanically combine them with the Mallin device to arrive at appellant's claimed combination, we find absolutely no basis for making such combination neither disclosed nor suggested in the patents relied upon. **In our view only appellant's specification suggests any reasons for combining the features of the secondary references with the primary reference and under the provisions of 35 U.S.C. 103 that does not constitute a bar.**

*Ex parte Fleischmann*, 157 U.S.P.Q. 155 (Bd. App. 1967). (Emphasis added.)

In the instant application, the examiner has done little more than cite references to show that one or more elements or subcombinations thereof, when each is viewed in a vacuum, is known. The claimed invention, however, is clearly directed to a combination of elements. That is to say, appellant does not claim that he has invented one or more new elements but has presented claims to a new combination of elements. **To support the conclusion that the claimed combination is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed combination or the examiner must**

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

**present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references.**

*Ex parte Clapp*, 227 U.S.P.Q. 972, 973 (Bd. App. 1985). (Emphasis added.)

In summary, as discussed more fully below, Claims 1-18 are clearly allowable in view of a lack of teaching or suggestion in the cited and applied references (or any obvious reason) of how they could be combined in any manner, much less in the manner recited in these claims.

Statements in the Office Action, such as "and this suggests some routines known in the art of debugging" (see Office Action, page 9), "hence has suggested routines similar to debugger calls" (see Office Action, page 10), and "one being reminiscent of debug calls" (see Office Action, page 10), are not teachings or suggestions of the type required by the foregoing decisions. They are evidence of effort to combine the references in a way to achieve the result of the claims in the invention. Nowhere do the cited and applied references expressly or implicitly suggest the combination claimed by the Office Action. And there is no convincing line of reasoning as to why one of ordinary skill in the art would have found the claimed invention to have been obvious in light of the teachings of the references. Furthermore, even if the references were combinable in the manner discussed in the remarks accompanying the rejection of these claims, which applicants specifically deny, the resultant combination would not meet all of the recitations of the claims, as discussed below.

**A. Independent Claims 1, 7, and 13**

The Office Action has failed to show, and applicants have been unable to find, where any of the cited and applied references teaches or even remotely suggests the subject matter of independent Claims 1, 7, and 13, the only independent claims in this application. Claim 1 is a method claim, Claim 7 is a computer-readable medium claim, and Claim 13 is a system claim. More specifically, Claim 1 is directed to a method for patching a computer application program including a plurality of executable steps; Claim 7 is directed to a computer-readable medium having computer-executable instructions for patching a computer application program including a plurality of executable steps; and Claim 13 is directed at a computer system for patching a computer application program wherein the computer system is capable of running an application having a plurality of executable steps. After amendment, Claims 1, 7, and 13 all recite:

determining whether or not the computer application program is compatible with a computer operating system executing the computer application program; and

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

if the computer application program is determined to be incompatible with the computer operating system ("incompatible application"), starting a debugger to run the incompatible application, the debugger:

setting at least one breakpoint within the incompatible application indicating a stopping point for the debugger;

running the steps of the incompatible application through the debugger;  
and

patching the incompatible application whenever a breakpoint has been reached.

This subject matter is not taught or even remotely suggested by either Straub or Nowlin. Neither teaches starting a debugger to run a computer application program in order to determine if the computer application program is or is not compatible with the computer operating system. Nor do they teach the debugger setting one or more breakpoints within an incompatible application and patching the incompatible application whenever a breakpoint has been reached.

The Office Action appears to allege that the claims of this application do not adequately define a debugger. See Office Action, pages 10 and 12. As a result, the Office Action broadly interprets this term, inappropriately in applicants' opinion. The common connotation of a debugger in the computer industry is a program designed to aid in the debugging of another program by allowing a programmer to step through the program, examine the data, and monitor conditions such as the values of variables. See *Microsoft Computer Dictionary*, 5th Edition. The use of the term debugger in the claims of this application is based on the common connotation that is prevalent in the computer software industry. On the contrary, neither Straub nor Nowlin teaches using such a debugger, expressly or implicitly. Furthermore, neither Straub nor Nowlin teaches a debugger setting one or more breakpoints within an incompatible application and patching the incompatible application whenever a breakpoint has been reached. As a result, neither Straub nor Nowlin teaches the debugging process recited in Claims 1, 7, and 13.

While the Office Action correctly concludes that Straub does not teach starting up a debugger upon determining an incompatibility between the application and the operating system, the Office Action incorrectly suggests that Nowlin provides a debugger process or something equivalent to a debugging process. See Office Action, page 3. Applicants respectfully disagree. Applicants have been unable to locate any pertinent subject matter in the portions of Nowlin (Col. 2, line 62, to Col. 3, line 39; Figures 2 and 3; and Col. 4, line 34, to Col. 5, line 42) referenced in the Office Action. The Office Action alleges that what Nowlin teaches in these portions of text is similar to a debugging process. In these portions of text, Nowlin teaches using

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

a surrogate translation layer and a filter process to enable a Win 9x application to run on a Windows® CE computer system. The surrogate translation layer converts calls for a non-Windows® CE operating system to calls that are compatible with a Windows® CE operating system. The surrogate translation layer contains a surrogate set of Win 9x kernel files such as kernel32.DLL, user32.DLL, gdi32.DLL, and others. The file filter resolves the differences between Win 9x executables and dynamic link libraries and the counterpart Windows® CE executables and the DLLs. The surrogate translation layer and the file filter do not constitute a debugger. Nor does the translation process or the filtering process constitute a debugging process. Further, nowhere does Nowlin suggest or mention a debugger or a debugging process. Applicants are unaware of a contemporary debugger that converts one type of API calls to another type of API calls. The Examiner has yet to show the applicants the existence of such a debugger.

The Office Action further alleges that Straub's approach in having calls invoked to look for specific properties in the code in order to establish incompatibility issues suggests some routines known in the art of debugging. See Office Action, page 9. Even if this allegation is true, which applicants categorically deny, it still does not remove the fact that Straub does not specifically teach using a debugger to identify an incompatible application, setting breakpoints in the incompatible application, and patching the incompatible application when a breakpoint is reached.

The Office Action also alleges that Nowlin suggested routines similar to a debugger call by evoking a collaboration of dynamic linked programs or conversion code to establish points at which incompatible data are appropriate for patching. See Office Action, page 10. Even if this allegation is true, which applicants categorically deny, Nowlin still fails to specifically teach using a debugger to identify incompatible applications and to set breakpoints within the incompatible application so patching can be done.

The Office Action further alleges that Nowlin's teaching provides effects of both debugging with location identification and corrective action with code modification. See Office Action, page 10. Again, this allegation is evidence of hindsight construction of the claimed invention. Even if Nowlin does provide location identification and code modification, Nowlin does not teach using a debugger that provides location identification and code modification.

In summary, nowhere do Straub and Nowlin teach using a debugger, much less a debugger having the functionality recited by Claims 1, 7, and 13.

The Office Action also alleges that Preisler teaches setting and monitoring for breakpoints and applying the patching process upon such breakpoints. See Office Action,

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

page 11. Applicants respectfully disagree. Claims 1, 7, and 13 all recite patching an incompatible application whenever a breakpoint has been reached. Even if a patch site is the equivalent of a breakpoint, as suggested by the Office Action and which the applicants categorically deny, in Preisler, patching does not occur whenever a patch site has been reached. Preisler teaches using a check command, or uncheck command, created by a user to indicate whether or not a particular patch site is to be patched.

Furthermore, there is no teaching or suggestion in Straub, Nowlin, and Preisler, taken alone or in combination, why it would be obvious to combine the individual teachings of these references. Thus, there is simply no teaching or suggestion in Straub, Nowlin, or Preisler of the subject matter recited in Claims 1, 7, or 13. Hence, applicants submit that Claims 1, 7, and 13 are clearly allowable.

Since all of the other claims remaining in this application depend from Claims 1, 7, and 13, respectively, these claims are submitted to be allowable for at least the same reasons that Claims 1, 7, and 13 are allowable. Further, these claims are submitted to be allowable for additional reasons.

**B. Dependent Claims 3, 9, and 15**

Claims 3, 9, and 15 are dependent upon Claims 1, 7, and 13, respectively. Each of Claims 3, 9, and 15 recites that determining if the application is compatible or incompatible with the operating system comprises (a) determining if at least the one identifying attribute of a plurality of identifying attributes of a computer application program matches at least one identifying attribute of a plurality of identifying attributes of incompatible applications; and (b) if at least one of the identifying attributes matches, determining that the computer application program is incompatible, otherwise determining that the computer application program is compatible.

The Office Action alleges that Straub teaches the subject matter of these claims. Applicants respectfully disagree. Straub teaches using a table consisting of the names of compatible computer programs and the version numbers of the operating systems installed in the personal computer. Straub teaches identifying attributes of a computer program against attributes of compatible computer programs of a computer system. Nowhere does Straub teach matching identifying attributes of a computer program against attributes of incompatible applications of an operating system.

More importantly, as noted above, even if it is obvious for one of ordinary skill in the art to modify the teaching of Straub to the claimed limitations in Claims 3, 9, and 15—which

applicants categorically deny—Straub's teaching still would not anticipate the subject matter of Claims 3, 9, and 15 when the subject matter of these claims is considered in combination with the subject matter of Claims 1, 7, and 13, the claims from which these claims depend. Consequently, applicants respectfully submit that Claims 3, 9, and 15 are allowable for reasons in addition to the reasons why Claims 1, 7, and 13 are allowable.

In addition, the Office Action alleges that the applicants failed to show specifics on how the rejection amounts to teaching away from the prior art or is at odds with the invention. See Office Action, page 12. Applicants respectfully disagree. Applicants do not have a duty to show specifically how the rejection amounts to teaching away from the prior art or is at odds with the invention in view of the rationales behind the rejections. Applicants have the duty to show how the Examiner's rejection fails to make a *prima facie* show of obviousness. The above discussion reveals that the Examiner has yet to make a *prima facie* showing of why applicants' invention is obvious in view of the cited references, while the cited references do not teach the subject matter recited by the claims and there is no suggestion or motivation to combine the cited references.

C. Dependent Claims 4, 10, and 16

Claims 4, 10, and 16 are dependent on Claims 3, 9, and 15, respectively. Each of Claims 4, 10, and 16 recites that determining if the application is compatible or incompatible with the operating system further comprises (a) storing identifying attributes of incompatible applications; and (b) retrieving at least one of the stored identifying attributes for determining if at least one identifying attribute of a plurality of identifying attributes of the computer application program matches at least one of the stored identifying attributes of the incompatible applications.

The Office Action alleges that Straub teaches the subject matter disclosed by these claims. Applicants respectfully disagree. As noted above, what Straub teaches is to store identifying attributes of compatible computer programs, such as program names, and the version numbers of the compatible operating systems of the personal computer. Nowhere does Straub teach storing identifying attributes of incompatible applications.

More importantly, as noted above, even if it is obvious for one of ordinary skill in the art to modify the teaching of Straub to the claimed limitations in Claims 4, 10, and 16, which applicants categorically deny, Straub's teaching still would not anticipate the subject matter of Claims 4, 10, and 16 when the subject matter of these claims is considered in combination with the subject matter of Claims 1, 3, 7, 9, 13, and 15, the claims from which these claims depend.

As a result, applicants respectfully submit that these claims are allowable for reasons in addition to the reasons why the claims from which these claims depend are allowable.

D. Dependent Claims 5, 11, and 17

Claims 5, 11, and 17 depend from Claims 1, 7, and 13, respectively. Each of Claims 5, 11, and 17 recites that setting at least one breakpoint within the incompatible application comprises (a) loading a debugger dynamic link library containing a list of breakpoints, each breakpoint having a handler having a set of instructions for patching the incompatible application; and (b) the debugger accessing the list of breakpoints from the debugger dynamic link library and setting the breakpoints within the incompatible application. The Office Action alleges that Claims 5, 11, and 17 are anticipated by the combined teaching of Straub, Nowlin, and Preisler. Applicants respectfully disagree.

The Office Action alleges that Straub and Preisler combined teach "executing a debugger containing a set or list of breakpoint . . . the debugger setting a set of breakpoints within the application." See Office Action, page 7. Applicants respectfully disagree. First, Claims 5, 11, and 17 recite a debugger dynamic link library, not the debugger itself, containing a list of breakpoints. Second, Straub teaches inserting one set of instructions so that the program image may run from a computer lower memory region. Nowhere does Straub teach using a dynamic link library containing a list of breakpoints, each breakpoint having a handler having a set of instructions for patching the incompatible application. Nor does Straub teach the use of a debugger setting the breakpoints within the incompatible application. Third, even assuming for purposes of argument that Preisler discloses executing a debugger containing a set of breakpoints, each breakpoint having a set of instructions for patching an application—which applicants categorically deny—this teaching does not teach the use of a dynamic link library. What Preisler does teach is to identify the patch sites and accumulating information regarding these patch sites in a table. Nowhere does Preisler mention the use of a dynamic link library containing a list of breakpoints, each breakpoint having a handle having a set of instructions for patching the incompatible application.

The Office Action further alleges that Nowlin teaches a dynamic link library. See Office Action, page 7. It is true that Nowlin teaches a file filter which is a dynamic link library. However, nowhere does Nowlin teach the dynamic link library being loaded by a debugger. Nor does Nowlin teach the dynamic link library containing a list of breakpoints, each breakpoint having a set of instructions for patching the incompatible application.

Furthermore, there is no teaching or suggestion in Straub, Nowlin, and Preisler, taken alone or in combination, why it would be obvious to combine the individual teachings of these references. More importantly, as noted above, even if these references were combinable—which applicants categorically deny—the resulting combination would not anticipate the subject matter of Claims 5, 11, and 17 when the subject matter of these claims is considered in combination with the subject matter of Claims 1, 7, and 13, the claims from which these claims depend. As a result, applicants respectfully submit that Claims 5, 11, and 17 are clearly allowable for reasons in addition to the reasons why the claims from which these claims depend are allowable.

E. Dependent Claims 6, 12, and 18

Claims 6, 12, and 18 depend from Claims 1, 7, and 13, respectively. Each of Claims 6, 12, and 18 recites that patching the application whenever a breakpoint has been reached comprises (a) calling the handler associated with the breakpoint; and (b) patching the incompatible application based on the instructions within the handler. As pointed out above, Straub does not teach using breakpoints. Nor does Straub teach calling a handler associated with a breakpoint or patching the incompatible application based on the instructions within the handler. Further, as pointed out above, Preisler does not teach patching upon reaching a patch site. In Preisler, a user determines whether to patch or not at a particular patch site. Furthermore, as pointed out above, Nowlin does not teach the use of a debugger DLL, nor the use of breakpoints, or a handler associated with a breakpoint.

Therefore, the cited references, taken alone or in combination, do not teach or even remotely suggest the subject matter of Claims 6, 12, and 18. Further, there is no teaching or suggestion in Straub, Nowlin, and Preisler, taken alone or in combination, why it would be obvious to combine the individual teachings of these references. More importantly, as noted above, even if these references were combinable, which applicants categorically deny, the resulting combination would not anticipate the subject matter of Claims 6, 12, and 18 when the subject matter of these claims is considered in combination with the subject matter of Claims 1, 7, and 13, the claims from which these claims depend. As a result, applicants respectfully submit that Claims 6, 12, and 18 are allowable for reasons in addition to the reasons why the claims from which these claims depend are allowable.

CONCLUSION

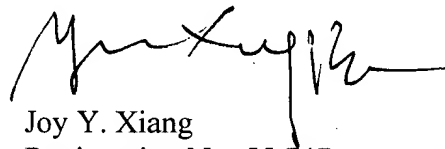
In view of the foregoing comments, applicants respectfully submit that all of the claims in this application are clearly allowable in view of the cited and applied references.

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100

Consequently, early and favorable action allowing these claims and passing this application to issue is respectfully solicited. If the Examiner has any questions, he is invited to contact applicants' attorney at the number set forth below.

Respectfully submitted,

CHRISTENSEN O'CONNOR  
JOHNSON KINDNESS<sup>PLLC</sup>



Joy Y. Xiang  
Registration No. 55,747  
Direct Dial No. 206.695.1607

JYX:nfs

LAW OFFICES OF  
CHRISTENSEN O'CONNOR JOHNSON KINDNESS<sup>PLLC</sup>  
1420 Fifth Avenue, Suite 2800  
Seattle, Washington 98101  
206.682.8100